UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/611,402	07/06/2000	Paul F. Ringseth	3382-56061	6516

7590 09/17/2003

Klarquist Sparkman Campbell Leigh & Whinston LLP
One World Trade Center
Suite 1600
121 S W Salmon Street
Portland, OR 97204

EXAMINER	
WOOD, WILLIAM H	
ART UNIT	PAPER NUMBER
2124	4

DATE MAILED: 09/17/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/611,402	RINGSETH ET AL.	
	Examiner	Art Unit	
	William H. Wood	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 June 2003* .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *06 July 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) *2 and 3* .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

Claims 1-25 have been examined.

Information Disclosure Statement

The information disclosure statements (IDS) submitted on 06 July 2000 and 02 June 2003 were considered by the examiner.

Drawings

The drawings submitted were approved by the draft person.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1 and 3-6 are rejected under 35 U.S.C. 103(a) as being unpatentable over "**C++ Builder** 5 Features & Benefits" by Jurgen Fesslmeier in view of **Corbett** et al. (USPN 5,515,536).

In regard to claim 1, **C++ Builder** disclosed the limitations:

- *In a computer system, a method of generating a interface implementation* (page 9, box "IDL Integration" and page 11, box "Integrated Type Library Creation Reduces the Number of Programming Tasks"), *the method comprising:*

- *receiving definition information* (page 9, box "IDL Integration", IDL is definition information) *interface features of a interface, interface including plural methods and one or more other methods* (page 9, box "IDL Implementation");
- *receiving programming language code for the one or more other methods, each of the one or more other methods having a name* (page 9, box "IDL Integration", C++ Implementation);
- *based upon the definition information and the programming language code, generating a interface implementation for operating the one or more other methods* (**C++ Builder** environment has IDL and C++ code), *interface implementation including:*
  - *executable code for the one or more other methods* (**C++ Builder** environment has IDL and C++ code);

C++ Builder did not explicitly state dispatch interface from the definition information. **Corbett** demonstrated that it was known at the time of invention to implement dispatch interfaces (column 5, lines 29-61); including executable code for mapping identifiers (column 5, lines 35-38); and executable code for calling the other methods (column 5, liens 41-61). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **C++ Builder**'s compiler system with dispatch operations as found in **Corbett's** teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide interface functionality for a common interface system of dispatches.

In regard to claim 3, **C++ Builder** and **Corbett** further disclosed the limitation *wherein a file includes the programming language code and a statement for importing the definition information* (**C++ Builder**: page 11, box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

In regard to claim 4, **C++ Builder** and **Corbett** further disclosed the limitation *wherein the generating comprises: in the second dispatch method implementation code, creating code for handling the arguments of one of the one or more other methods with a generic data structure* (**Corbett**: column 5, lines 35-38).

In regard to claim 5, **C++ Builder** and **Corbett** further disclosed the limitation *wherein the dispatch interface implementation is part of a dual interface implementation, the method further comprising: generating executable code for directly invoking the one or more other methods through a vtable mechanism at run time* (**Corbett**: column 5, lines 35-38).

In regard to claim 6, **C++ Builder** and **Corbett** further disclosed the limitation *wherein the dispatch interface implementation further includes:*

- *executable code for a third dispatch method, the third dispatch method for determining the availability of type information for the dispatch interface* (**Corbett**: column 5, lines 62-67)*; and*

- *executable code for a fourth dispatch method, the fourth dispatch method for retrieving available type information for the dispatch interface* (**Corbett**: column 5, lines 62-67).

Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over "**C++ Builder** 5 Features & Benefits" by Jurgen Fesslmeier in view of **Corbett** et al. (USPN 5,515,536) as applied to claim 1 and in further view of **Yellin** et al. (USPN 5,946,489).

In regard to claim 2, **C++ Builder** and **Corbett** did not explicitly state the limitation *wherein the definition information is embedded in a file for the programming language code*. **Yellin** demonstrated that it was known at the time of invention to place code of one type within a file of a differing type of code (column 8, lines 21-26; inlining). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the compiling system of **C++ Builder** and **Nakamura** with inlining (of IDL) as suggested by **Yellin**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide as much information in a single and thus readily available source. Furthermore, **C++ Builder** implied the definition information could be embedded in the programming language code file (**C++ Builder**: page 11, box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

Claims 7-11, 13-16 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over "**C++ Builder** 5 Features & Benefits" by Jurgen Fesslmeier in view of **Nakamura** et al. (USPN 5,987,529).

In regard to claim 7, **C++ Builder** disclosed the limitation

- *A computer readable medium having stored thereon a computer executable compiler system that generates a implementation from definition information and programming language code* (page 9, box "IDL Integration"), *the compiler system comprising:*
  - *a front end module that receives definition information and programming language code, the definition information defining interface features of a interface, the programming language code for implementing one or more methods* (page 9, box "IDL Integration", IDL is definition information and C++ Implementation);
  - *a converter module that identifies relations between the definition information and the one or more methods* (**C++ Builder** environment has IDL and C++ code); *and*
  - *a back end module that generates a interface implementation based upon the relations, the interface implementation for operating the one or more methods* (page 9, box "IDL Integration" and page 11, box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

C++ Builder did not explicitly state late bound. **Nakamura** demonstrated that it was known at the time of invention to late bound interfaces and methods (column 1, line 15 to column 4, line 35). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **C++ Builder**'s system of programming using integrated IDL with the ability to create interfaces for late bound situations as found in **Nakamura**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to develop interfaces for as many situations as possible, including late bound. Furthermore, **C++ Builder** illustrated the use of COM interfaces (page 11, box "COM+ Server Components" and "Built in COM+ Support" and box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

In regard to claim 8, **C++ Builder** and **Nakamura** disclosed the limitation *wherein the converter module identifies one or more relations, each relation between one of the one or more late bound methods and a corresponding identifier, and wherein based upon the one or more relations the back end module generates for each of the one or more late bound methods code mapping the name of the late bound method to the corresponding identifier for the late bound method* (**Nakamura**: column 1, lines 37-50).

In regard to claim 9, **C++ Builder** and **Nakamura** disclosed the limitation *wherein the converter module identifies one or more relations, each relation between one of the one or more late bound methods and a corresponding identifier, and wherein based upon the one or more relations the back end module generates for each of the one or more*

late bound methods code for calling the late bound method upon receipt of the corresponding identifier for the late bound method (**Nakamura**: column 1, lines 37-50).

In regard to claim 10, **C++ Builder** and **Nakamura** disclosed the limitation *wherein the converter module identifies one or more relations, each relation between type information and an argument of one of the one or more late bound methods, and wherein based upon the one or more relations the back end module generates code for handling the arguments of the late bound method with a generic data structure* (column 1, lines 28-50).

In regard to claim 11, **C++ Builder** and **Nakamura** disclosed the limitation *wherein the converter module identifies a relation between a property indicator and one of the one or more late bound methods, and wherein based upon the relation the back end module generates code for retrieving or setting a corresponding property through the late bound method* (**Nakamura**: through use of dispatch table; and **C++ Builder**: COM).

In regard to claim 13, **C++ Builder** disclosed the limitations:

- *A computer readable medium having stored thereon computer executable instructions for performing a method of automatically generating a interface implementation* (page 9, box "IDL Integration"), *the method comprising:*
  - *receiving programming language code for one or more methods of a late binding interface* (page 9, box "IDL Integration", C++ implementation);

- *receiving definition information that defines interface features of the late binding interface* (page 9, box "IDL Integration", IDL is definition information);
- *based upon the programming language code and the definition information* (**C++ Builder** environment has IDL and C++ code), *generating a interface implementation for operating the one or more methods, the interface implementation including one or more methods, a first method for calling the one or more methods responsive to client requests* (page 9, box "IDL Integration" and page 11, box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

C++ Builder did not explicitly state late bound. **Nakamura** demonstrated that it was known at the time of invention to late bound interfaces and methods (column 1, line 15 to column 4, line 35). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **C++ Builder**'s system of programming using integrated IDL with the ability to create interfaces for late bound situations as found in **Nakamura**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to develop interfaces for as many situations as possible, including late bound. Furthermore, **C++ Builder** illustrated the use of COM interfaces (page 11, box "COM+ Server Components" and "Built in COM+ Support" and box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

In regard to claim 14, **C++ Builder** and **Nakamura** disclosed the limitation *wherein the first late binding method lacks a call to a separate late binding interface implementation* (**Nakamura**: the rejection of claim 15 is incorporated herein, for the first late bound method).

In regard to claim 15, **C++ Builder** and **Nakamura** disclosed state the limitation *wherein a second late binding method maps names of the one or more late bound methods to corresponding identifiers for run time binding, and wherein the second late binding method lacks a call to a separate late binding interface implementation.* **Nakamura** demonstrated that it was known at the time of invention to utilize a dispatch table, which is the functionality described above (column 1, lines 15-50).

In regard to claim 16, **C++ Builder** and **Nakamura** disclosed the limitation *wherein the late binding interface implementation includes a second late binding method for determining the availability of type information, and wherein the late binding interface implementation further includes a third late binding method for retrieving available type information.* However, this limitation is found in the same manner as claim 15, the rejection being incorporated herein using **Nakamura**.

In regard to claim 18, **C++ Builder** and **Nakamura** disclosed the limitation *wherein the generating comprises: identifying type information for an argument of a first late bound method; for the implementation for the first late binding method, generating code for*

handling the argument with a generic data structure. However, this limitation is found in the same manner as claim 15, the rejection being incorporated herein using **Nakamura**.

Claim 17 is rejected under 35 U.S.C. 103(a) as being unpatentable over "**C++ Builder** 5 Features & Benefits" by Jurgen Fesslmeier in view of **Nakamura** et al. (USPN 5,987,529) as applied to claim 13 and in further view of **Yellin** et al. (USPN 5,946,489).

In regard to claim 17, **C++ Builder** and **Nakamura** did not explicitly state the limitation *wherein the definition information is embedded in a file for the programming language code.* **Yellin** demonstrated that it was known at the time of invention to place code of one type within a file of a differing type of code (column 8, lines 21-26; inlining). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the compiling system of **C++ Builder** and **Nakamura** with inlining (of IDL) as suggested by **Yellin**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide as much information in a single and thus readily available source.

Claims 12 and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over "**C++ Builder** 5 Features & Benefits" by Jurgen Fesslmeier in view of **Nakamura** et al. (USPN 5,987,529) as applied to claims 7 and 13 and in further view of **De Groot** et al. (USPN 5,842,220).

In regard to claim 12, **C++ Builder** and **Nakamura** did not explicitly state the limitation *wherein the late binding interface implementation is part of a combined early binding and late binding interface implementation, and wherein the back end module further generates an early binding interface implementation for the one or more late bound methods.* **De Groot** demonstrated that it was known at the time of invention to utilize late and early bound interfaces and methods (column 1, lines 42-55; column 3, lines 18-45; column 4, lines 35-39; and column 10, line 41-56). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **C++ Builder**'s system of programming using integrated IDL with the ability to create interfaces for early and late bound situations as found in **De Groot**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to develop interfaces for as many situations as possible, including early bound (furthermore, **De Groot** mentions providing early bound interfaces specifically, column 10, lines 41-56). Furthermore, **C++ Builder** illustrated the use of COM interfaces (page 11, box "COM+ Server Components" and "Built in COM+ Support" and box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

In regard to claim 19, **C++ Builder** and **Nakamura** did not explicitly state *wherein the late binding interface implementation adjoins an early binding interface implementation, the method further comprising: generating the early binding interface implementation for directly invoking the one or more late bound methods.* **De Groot** demonstrated that it was known at the time of invention to utilize late and early bound interfaces and

methods (column 1, lines 42-55; column 3, lines 18-45; column 4, lines 35-39; and column 10, line 41-56). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **C++ Builder**'s system of programming using integrated IDL with the ability to create interfaces for early and late bound situations as found in **De Groot**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to develop interfaces for as many situations as possible, including early bound (furthermore, **De Groot** mentions providing early bound interfaces specifically, column 10, lines 41-56). Furthermore, **C++ Builder** illustrated the use of COM interfaces (page 11, box "COM+ Server Components" and "Built in COM+ Support" and box "Integrated Type Library Creation Reduces the Number of Programming Tasks").

Claims 20-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over "**C++ Builder** 5 Features & Benefits" by Jurgen Fesslmeier in view of **De Groot** et al. (USPN 5,842,220).

In regard to claim 20, **C++ Builder** disclosed the limitation:

- *In a computer system, a method of automatically generating an interface implementation* (page 9, box "IDL Integration"), *the method comprising:*
  - *receiving programming language code; for one or more methods of an interface* (page 9, box "IDL Integration", C++ implementation);

- *receiving definition information that defines interface features of the interface* (page 9, box "IDL Integration", IDL is definition information);
- *based upon the programming language code and the definition information, generating an interface* (**C++ Builder** environment has IDL and C++ code; page 9, box "IDL Integration" and page 11, box "Integrated Type Library Creation Reduces the Number of Programming Tasks"),

C++ Builder did not explicitly state early or late bound. **De Groot** demonstrated that it was known at the time of invention to utilize late and early bound interfaces and methods (column 1, lines 42-55; column 3, lines 18-45; column 4, lines 35-39; and column 10, line 41-56). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **C++ Builder**'s system of programming using integrated IDL with the ability to create interfaces for early and late bound situations as found in **De Groot**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to develop interfaces for as many situations as possible, including early and late bound (furthermore, **De Groot** mentions providing early bound interfaces specifically, column 10, lines 41-56). The limitations *wherein the early binding mechanism provides for direct invocation of the one or more methods* (**De Groot**: column 10, line 41-56), *and wherein the late binding mechanism provides for invocation of the one or more methods responsive to a request through a late binding method* (**De Groot**: column 1, line 42-55) are disclose as well.

In regard to claim 21,**C++ Builder** and **De Groot** disclosed the limitation *wherein the late binding mechanism maps names of the one or more methods to corresponding identifiers at run time* (**De Groot**: column 1, lines 46-49).

In regard to claim 22, **C++ Builder** and **De Groot** disclosed the limitations *wherein the generating comprises: identifying type information for an argument of a first method; for the late binding mechanism, creating code for handling the argument with a generic data structure* (**De Groot**: column 1, lines 46-49).

In regard to claim 23, **C++ Builder** disclosed the limitations:

- *In a computer system, a method of automatically generating client side call site code* (page 9, box "IDL Integration"), *the method comprising:*
  - *receiving definition information for interface features of a interface* (page 9, box "IDL Integration", IDL is definition information);
  - *receiving programming language code for calling a method of the interface* (page 9, box "IDL Integration", C++ implementation);
  - *based upon information for one or more input arguments of the method, generating code* (**C++ Builder** environment has IDL and C++ code; page 9, box "IDL Integration" and page 11, box "Integrated Type Library Creation Reduces the Number of Programming Tasks")

C++ Builder did not explicitly state type information and late bound and code *for packing the one or more arguments into a generic argument data structure; and*

generating code for calling the late bound method through an invocation method of the late binding interface, wherein the calling includes passing the generic argument data structure to the invocation method. **De Groot** demonstrated that it was known at the time of invention to utilize late bound interfaces and methods (column 1, lines 41-55); to utilize packing and unpacking from a data structure (column 1, lines 46-49); and type information (column 1, lines 42-55). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **C++ Builder**'s system of programming using integrated IDL with the ability to create interfaces for late bound situations; packing and unpacking; and type information as found in **De Groot**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to develop interfaces for as many situations as possible, including late bound, especially when those interfaces are needed for so many applications.

In regard to claim 24, **C++ Builder** and **De Groot** further disclosed the limitation *further comprising: based upon type information for a return value of the late bound method, generating code for unpacking the return value from a generic return value data structure* (**De Groot**: column 1, lines 42-55).

In regard to claim 25, **C++ Builder** and **De Groot** further disclosed the limitation *further comprising: generating code for calling a mapping method of the late binding interface, the mapping method associating a late bound method name with an identifier* (**De Groot**: column 1, lines 46-49).

Correspondence Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to William H. Wood whose telephone number is (703)305-3305. The examiner can normally be reached 7:30am - 5:00pm Monday thru Thursday and 7:30am - 4:00pm every other Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662. The fax phone numbers for the organization where this application or proceeding is assigned are (703)746-7239 for regular communications and (703)746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703)305-3900.

William H. Wood
September 2, 2003

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100